UNIVERSITY OF QUEENSLAND
COMPUTER CENTRE

COMPUTER CENTRE BULLETIN

Vol. 4, No. 4.
5 April 1971

Editor:
Mrs Sarah Barry

COMPUTER SYSTEM STATUS SERVICE

The automatic telephone answering service that gives the Computer System Status (on extension 8101) has now been operating for some time and it appears that users have made good use of it.

In the event of a system crash, the Centre requests users to wait about 10 minutes before phoning the Service. This delay gives the Centre's staff a chance to try to determine the cause of the crash, decide on what remedial action is necessary, form some estimate of the likely time of resumption of service and record a suitable message.

It must be emphasized that times given for resumption of service after a fault are estimates only. When any piece of complex equipment breaks down, it is impossible to say exactly how long it will take to trace the fault and effect a repair. A computer is no exception. Thus, as work progresses, the estimated time of availability will be revised.

In these circumstances, users are requested to contact the Status Service before attempting to log in again and check on availability.

EXTENSION OF TIMESHARING HOURS

From Monday 22 March the hours of timesharing operation were extended to cover the normal hours of operation of the PDP-10 system.

The Centre will no longer announce the hours of timesharing and batch operation separately, but will simply publish hours of 'system availability' for the PDP-10. These hours will cover both modes of processing.

It will be necessary for the Computer Centre staff to reserve the system periodically for dedicated development testing. The Centre will attempt to make such reservations so as to inconvenience the least number of users. Reservations will be notified via the Computer System Status Service, the Schedule Board and the log-in notice, with as much advance warning as possible being given.

FILE LISTING SERVICE FOR REMOTE TERMINAL USERS

From Monday 8 March the Computer Centre has provided a service to list files of remote terminal users on the line printer. This operates on a similar basis to the existing service for loading card decks to file.

Users who wish to have files listed should give a written request to the Administrative Officer that specifies:

(a) User's name

(b) Project number

(c) Password

(d) Names of files to be listed

Terminal users who wish to have output from FORTRAN programs listed on the line printer should write their output to a named disk file. This output should include all the necessary carriage control characters. The written request for such files to be listed must indicate that the file was output from FORTRAN by the notation 'FORTRAN OUTPUT' against the filename.

Listings will be placed in the PDP-10 output racks.

This is an interim service only. It will be discontinued immediately an alternative means for users to list their own files becomes available.

Charges at the standard rates will be made for this service.

RANDOMIZE COMMAND IN BASIC

The following letter has been received regarding the RANDOMIZE command in BASIC.

> It might be worthwhile drawing the attention of other users to the dangers of using this command more than once in generating a random sequence. An example like the attached could occur by accident if one slipped up on inserting the command inside instead of outside a loop - or it could occur if one were tempted to 'improve' the sequence of pseudo-random numbers by a periodic fresh start.
>
> Allan Beattie

```
.BASIC<cr>
NEW OR OLD--OLD<cr>
OLD FILE NAME--RAN<cr>

READY
LIST<cr>

RAN            14:58                11-MAR-71

1ØØ FOR I = 1 TO 2Ø
11Ø RANDOMIZE
12Ø PRINT RND
14Ø NEXT I
9ØØ END

READY
RUNNH<cr>

Ø.932586
Ø.932586
Ø.182Ø64
Ø.182Ø64
Ø.182Ø64
Ø.136758
Ø.218742
Ø.218742
Ø.218742
Ø.787849
Ø.887728
Ø.316491
Ø.316491
Ø.316491
Ø.369822
Ø.369822
Ø.665148
Ø.665148
Ø.269Ø1

TIME:  Ø.2Ø SECS.

READY
SYSTEM<cr>

EXIT
↑C

.
```

In BASIC, the random number generator is initialized using the time of day. Thus successive, closely executed RANDOMIZE statements will produce the same sequences as demonstrated by the example.

This is not really a BASIC error, but a misuse of the RANDOMIZE statement. As is stated by Mr. Beattie, the use of multiple initializations of a random number generator is not intended to 'improve' the sequence of numbers produced.

FORTRAN COURSE

The next Computer Centre FORTRAN programming course will be held immediately after Easter.

In a break with past tradition, this course will be held on Tuesday and Thursday evenings from 7 p.m. to 10 p.m. over a period of three weeks.

The course will commence at 7 p.m. on Tuesday, 13 April and will be held in lecture room B18 of the Engineering Administration Building.

The standard fee will be charged for this course. Nomination forms are available from the Computer Centre Secretary.

GE-225 FORTRAN IV

(a) There have been reported some examples of improper calculations and output conversions involving FORTRAN IV real quantities.

Investigation has shown that a problem exists, but that it occurs only intermittently. A hardware error was found to be part of the problem and this has now been remedied, but there appear to be other defects that have not yet been isolated.

So far, this problem has been seen only in the execution of FORTRAN programs and, if you have any demonstrable examples of this nature, we would like to see them, so that we can attempt to locate the problem area.

Users running FORTRAN IV programs on the GE-225 must be aware of this problem until it has been solved.

(b) A statement of the form

```
IF (logical expression) WRITE (I1,35Ø)
```

while correct, is not accepted by the compiler and error message 17 is issued.

It is possible to program around this situation by using the inverse of the logical expression within the logical IF, and a GO TO statement as in the following example.

```
         IF (.NOT.logical expression) GO TO 210
         WRITE (I1,350)
     210 CONTINUE
```

(c) Documentation Error

The execution error LBF, that is noted as occurring in the routine DMOD, can also occur in a number of other places. The error message actually originates in the absolute suffix routine FXS that converts a real format number to integer. It will occur if the real number to be converted is too large for integer representation. The message may thus occur as a result of calls to a number of library routines, for example, IDINT, DMOD, IFIX, INT, AINT, as well as implied calls to the absolute suffix routine.

PDP-10 FORTRAN IV

(a) Error in Complex Arithmetic

It has been found that in certain cases the FORTRAN IV compiler does not correctly translate expressions involving complex division. The problem occurs when the complex variable on the left hand side of an expression is also used as the divisor on the right hand side.

example:

```
         COMPLEX Z, Z1
         .
         .
         .
         Z=Z1/Z
```

During execution, this produces an error message

```
         ILL MEM REF AT USER adr
```

The problem can be avoided in the following manner, and the correct results produced.

```
         COMPLEX Z, Z1, Y
         .
         .
         .
         Y=Z
         Z=Z1/Y
```

This problem has been reported to Digital Equipment for correction.

(b) Representation of Characters on Coding Sheets

Users wishing to submit data preparation work to the Centre should use the standard Computer Centre coding sheets. At the top of each sheet is stated clearly the standard representation for certain characters, among them the letter O and the digit 0. On the coding sheets, the letter O is represented by θ, and the digit 0 is represented by 0 or Ø.

It has been pointed out that the examples throughout the PDP-10 FORTRAN manual do not correspond to this. They are not meant to. Instead, they correspond to the way in which they would be produced either on the line printer or on the Teletypes, where the letter O is left uncrossed and the digit 0 is slashed.

(c) Use of the Quote Character Within a Hollerith String

The quote character (') may be used within format statements as part of a Hollerith string in either of the following two ways:

(i) 9HSTUDENT'S

(ii) 'STUDENT''S'

Both of these usages will result in the printing of the same message, namely

```
STUDENT'S
```

It has been found that usage (i) when immediately preceded by the slash specification separator (/), is improperly handled, for example

```
1Ø  FORMAT (... /9HSTUDENT'S ...)
```

This error has been reported to DEC for rectification and the FORTRAN operating system will be corrected when their patch is available. In the mean time, either arrange format statements so that this situation does not occur, or use method (ii) to include a quote character within a Hollerith format specification.

DEVELOPMENT WORK IN CENTRE

Since the release of remote terminals, the Centre has received a number of requests for particular facilities and queries on future plans.

In order to help answer a number of these points, it has been decided to publish regularly, details of development work that Computer Centre staff are carrying out.

At the time of writing (15 March 71), staff are working on version two of the Timesharing System which involves modifications to the Operating System, extensions to the command set and development of a number of special system programs. Additionally, this will necessarily involve extensive documentation and introduce some new procedures for the Centre's operational staff.

Version two is scheduled for operation in the second quarter of this year. It will provide the following new facilities:

(a) Digital plotting

(b) Teletype paper tape I/O

(c) Access to another user's files as permitted by that user

(d) An extended command set to access a number of new systems programs

(e) A group accounting system that will enable, for example, large numbers of student jobs to be run under the one project number with individual cost limits and accounting.

(f) Ability to set cost limits for an individual task or group of tasks within a job.

More details of these services will be published as they become available.

A new version of Batch will also become available during the second quarter of the year. This will provide the batch user with many of the facilities available via remote terminals. In particular, the batch user will be able to store binary program files within the system and simply recall them for running as required.

On completion of the above, work will be concentrated on

(a) Provision for remote terminals to initiate the printing of files on the line printer

(b) A File Management System. This work should enable a further increase in file storage limits to be made. It is also hoped to provide facilities whereby a user will be able to submit his own magnetic tape for files to be input to, or dumped from, the system.

(c) The establishment of program libraries including statistical packages

The priorities assigned to PDP-10 developments are based upon user requests. Although these priorities may be modified by internal considerations, such as problems of overall system efficiency, it is vital that users continue to request particular developments; please direct all requests to the Director of the Centre.

SOFTWARE CLASSIFICATION

The Computer Centre will, in the near future, classify programs it distributes to users into one of four categories. This applies to both systems and applications software. 'Applications' covers areas that require no specialized knowledge of the operating system beyond that which is available to a normal user. The purpose of this classification system is to increase the availability of software to users and to specify the level of support that can be given for items of software.

Type 1 and Type 2 programs are programs that have been subjected to formal testing by the Computer Centre. Type 1 is the classification applied to systems software, and Type 2 applies to applications areas.

Type 3 programs are those that have been submitted by users, DECUS, etc., or that have been obtained from other sources. They are programs of general interest that are available for use by all users. Type 3 programs have met a basic set of programming and documentation standards, but have not been subjected to rigorous formal testing by the Computer Centre. The user is expected to make the final evaluation as to the usefulness of such programs in his own environment.

Type 4 programs are those contributed for general use from a variety of sources including users, DECUS, etc. They are made available by the Computer Centre essentially in the author's original form, but conform to the Computer Centre's published Type 4 standards (these may be virtually zero). The Computer Centre exercises no control over the technical content of the documentation, but merely distributes to interested parties the supplied information. Type 4 programs have not been tested by the Computer Centre.

Availability of programs under this classification scheme will be advised through future issues of the Bulletin.

LIBRARY ACCESSIONS

BEER, Stafford	*Cybernetics and management* 1967 (001.53 BEE Main)
MIDDLETON, Robert Gordon	*Computers and artificial intelligence* 1969 (001.535 MID Main)
BAKEWELL, Kenneth Graham Bartlett	*How to find out: management and productivity* 1970 (016.6585 BAK Main Ref.)
	Systems Analysis in Libraries 1969 (025.1015199 SYS Main)
DALLENBACH, Hans G.	*User's guide to linear programming* 1970 (519.92 DAE Main)

	International journal of computer mathematics no 3; 1970 and onwards (510.78 INT Maths)
	Handbook of data processing management 1970 (651.8 HAN Main)
KENNEDY, Michael 1939-	*Ten statement Fortran plus Fortran IV for the IBM featuring the WATFOR and WATFIV compilers* 1970 (651.8 KEN Engin)

A.C.S. DECISION TABLES SEMINAR

The Queensland branch of the Australian Computer Society held its second Professional Development Seminar on 12 February 1971. The seminar was concerned with the techniques of programming using 'Decision Tables' and papers were presented that covered not only introductory material but also details of applications and the implementation problems of decision table processing systems. The paper that follows has been reproduced from the proceedings of the seminar and serves as an introduction to the use of decision tables. A second paper, to be published in the next Bulletin, will describe some of the implementation features of decision table software. Both papers are reprinted by kind permission of the Australian Computer Society and the authors concerned.

INTRODUCTION TO DECISION TABLES

D.F. Abercrombie

David Abercrombie graduated from the University of Queensland in 1963 with a Bachelor of Science degree. Since 1964 he has worked with IBM as a Programmer, Systems Analyst and currently as Systems Programmer for IBM's 360/40 installation in Brisbane.

ABSTRACT

The decision table technique is finding increasing acceptance in the Data Processing community in the areas of Systems Analysis, Programming and Documentation.

This paper is intended to serve as a general introduction to the subject. Decision table 'jargon' is documented and conventions attaching to the use of the tables are outlined.

INTRODUCTION

The advantages of tabular presentation of data over the narrative form are well known. They may be summarized as follows:

(a) Conciseness

Tabular presentation eliminates unnecessary verbiage. Large volumes of *facts* can be presented in a single well-planned table.

(b) Preciseness

Tabular presentation imposes on the user a regular form that, when correctly employed, highlights deficiencies or redundancies during the preparation of a table.

(c) Clarity

Tables are easily understood and provide a convenient means of communication between people of different backgrounds and training.

DEFINITION

Most tables in everyday use demonstrate one of the principles basic to Decision Tables - the 'if-then' relationship. Consider the Income Tax table shown in Figure 1. The method of use of such a table may be epxressed thus: *If* a man's income is $10,000 p.a., *then* his tax deduction will be $3,000. The Income Tax is an example of a Data Table. If we add columns to the Income Tax table for the various levels of Dependants Deductions, then we have a more complex table (a compounding of the 'if' or condition, section of the table), but it is still a Data Table.

ANNUAL INCOME	TAX PAYABLE
$1000.00	$250.00
$2000.00	$600.00
$3000.00	$920.00
$4000.00	$1250.00
$5000.00	$1400.00
$6000.00	$1575.00
$7000.00	$1800.00
$8000.00	$2100.00
$9000.00	$2500.00
$10000.00	$3000.00

Figure 1 Income Tax Table

What distinguishes a Decision Table from a Data Table is that in a Data Table the function (i.e., what we get from the table) is data, while in a Decision Table the function is an action or series of actions to be performed.

A Decision Table is a tabular representation of the relationship between a set of logical conditions and a set of actions. It expresses the combinations of actions which are to be taken for each combination of conditions.

Figure 2 shows a simple example of a Decision Table.

Order quantity over limit?	Y	N	N	N				
Is client's credit O.K.?	-	Y	Y	N				
Sufficient stock on hand	-	Y	N	-				
Complete shipping order		X						
Send to despatch		X						
Back order			X					
Mark order 'Reject-Limit'	X							
Mark order 'Reject-Credit'				X				
Return order	X			X				

Figure 2 Decision Table

DECISION TABLE COMPONENTS

A Decision Table has four basic parts. Firstly there are two sections, the Condition Section and the Action Section (representing the components of the 'IF-THEN' relationship). On a Decision Table coding form, the two sections are separated by a double horizontal line (see Figure 3).

CONDITION SECTION
ACTION SECTION

Figure 3 Decision Table Components

Within each of the two sections, we have the Stub portion and the Entry portion. These are separated by a double vertical line on the coding form.

On the Stub, all possible conditions and resultant actions are listed. Entries are made in the Entry section of the table to define the valid combinations of conditions, and the action or set of actions to be taken for each such combination.

The letters 'Y' (for yes) and 'N' (for no) are coded in the Condition Entry cells of a table to indicate the existence or otherwise of a condition (blank denotes that the condition is not applicable). An 'X' is coded in an Action Entry cell if the action is to be executed.

The four basic components of a Decision Table are indicated in Figure 4.

Condition Stub	Condition Entries
Action Stub	Action Entries

Figure 4 Decision Table Components

Other terms used to describe characteristics of Decision Tables are as follows:

<u>Table Header</u>	Placed at top of Decision Table form as identification, e.g., Table number or description.
<u>Statement</u>	The combination of Stub and Entry on any row of the table - Condition Statement, Action Statement.
<u>Rule</u>	Each column of a table in the Entry section contains a Rule, i.e., a single combination of conditions and the set of actions to be performed for that combination.
<u>Rule Header</u>	Identification for each rule, e.g., Rule number.
<u>Row Header</u>	Identification for a particular row, e.g., Row number.

Figure 5 shows a fully labelled Decision Table form (see page 48).

<u>TYPES OF DECISION TABLES</u>

There are three types of Decision Tables, viz:

(a) Limited Entry Tables

(b) Extended Entry Tables

(c) Mixed Entry Tables

(a) <u>Limited Entry Tables</u> (Figure 6)

In a Limited Entry Table, the Stub portion of each row is a complete statement, and the entry portion may contain only Y, N or blank in the Condition section, X or blank in the Action section.

Card Code is '1'?	Y	N	N	N
Card Code is '2'?	-	Y	N	N
Card Code is '3'?	-	-	Y	N
Do Routine 1	X			
Do Routine 2		X		
Do Routine 3			X	
Do Error Routine				X

Figure 6 <u>Limited Entry Table</u>

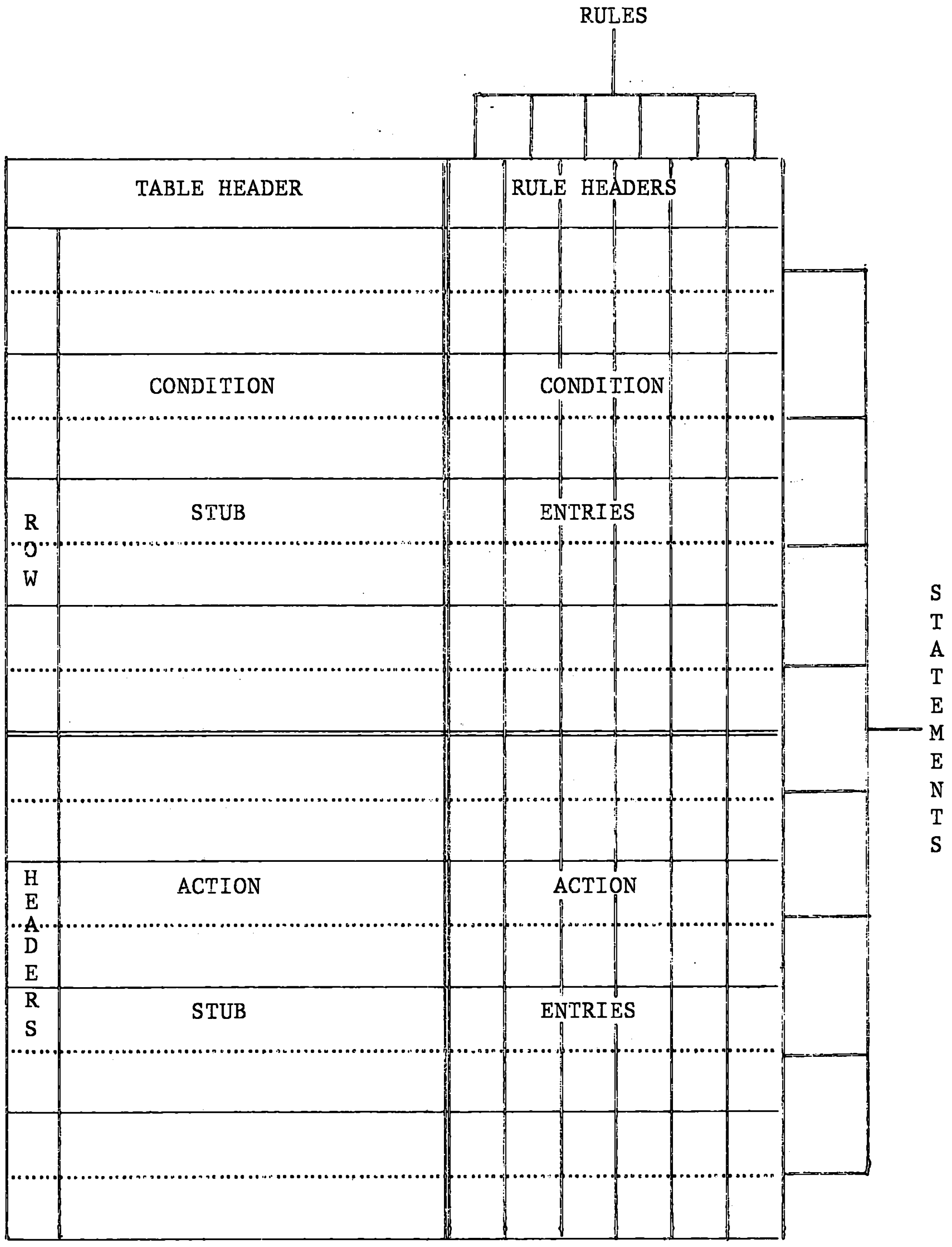

Figure 5 Decision Table

(b) Extended Entry Tables (Figure 7)

Here the Stub portions are incomplete statements that are completed by the contents of the Entry portions. Extended Entry Decision Tables can obviously be more compact than the Limited Entry type.

Card Code is -	1	2	3	Other
Do -	Routine 1	Routine 2	Routine 3	Error Routine

Figure 7 Extended Entry Table

(c) Mixed Entry Tables (Figure 8)

Mixed Entry Tables are simply a combination of Limited Entry and Extended Entry rows, e.g., the Condition section of a table may be in Limited Entry form, the Action section in Extended Entry form.

Card Code is '1'?	Y	N	N	N
Card Code is '2'?	-	Y	N	N
Card Code is '3'?	-	-	Y	N
Do -	Routine 1	Routine 2	Routine 3	Error Routine

Figure 8 Mixed Entry Table

PREPARATION OF DECISION TABLES

The best method of approach to the preparation of a Decision Table is to code the table first in limited entry form, then consolidate the table if possible to a mixed or extended entry form. As expertise in the technique is gained, the user may find that he is able to code directly in mixed or extended form without loss of accuracy. The following rules are recommended for the coding of the tables:

(a) Code the condition section first, beginning with tests that must always be made and ranging down to tests that are made only in special circumstances. Within this framework, tests should be listed in order of their likelihood of being true.

(b) Tests for each of a mutually exclusive set of conditions should be placed together, starting with the test most likely to be true, as above.

(c) The condition entries should be coded systematically, beginning with the first condition affirmative in the first rule. Consider the other conditions in turn. If a condition could affect the action to be taken on the rule in question, then enter 'Y', otherwise code '-'.

If the last rule written contains no 'Y' entries, then proceed to complete the action section (d). Otherwise, take the next rule, copy the previous rule from the top until the last 'Y' entry is reached, and code 'N' beside this in the new rule. Then return to consider the as-yet uncoded conditions as above.

(d) List the actions to be taken in the order in which they will be executed, as far as possible. Fill in the Action Entries for each rule.

(e) The last action statement for each rule should give an instruction as to where control is to be passed to on completion of the actions for that rule (i.e., a 'GO TO' statement).

(f) Check the table for completeness and possibility of condensation (see next Section).

Do not try to squeeze too much into the one table. Complex situations should be broken down into a number of simple tables, linked by 'do Table X' (closed table), or 'go to table Y' (open table) action statements.

TABLE OPTIMIZATION AND VALIDATION

(a) Rule Combination:

In some cases, more than one rule may have generated the same set of actions. It may be possible to combine the rules.

Two rules that have the same action entries and the same condition entries for all conditions but one may be replaced with a single rule having a '-' entry for the condition on which the two original rules had different entries.

(b) Completeness Check:

In a limited entry table, the total number of possible combinations of conditions is 2^N where 'N' is the number of conditions.

The number of combinations included in each rule is 2^M where 'M' is the number of '-' entries in that rule. A simple check on the completeness of a table is then that the sum of the number of combinations covered by each rule must be equal to 2^N.

i.e., $\Sigma 2^{Mi} = 2^N$, where Mi is the number of '-' entries in rule 'i'.

APPLICATION

Decision tables then find their application in situations where a number of conditions interact to determine courses of action to be followed. They provide a clear and concise medium for communication between persons with different views of the problem, in a familiar form.

REFERENCES

'Decision Tables - A Systems Analysis and Documentation Technique' - IBM publication F20-8102.

N.C.C. Systems Analysis Course Outline for Decision Tables.

'Programming and Documentation Standards using Decision Tables' - David F. Hunter (IBM Australia Limited) New Zealand Computer Conference, Dunedin, August 1970.